# *High Performance Computing* Cluster Quick Reference User Guide

## Base Operating System:

*Redhat(TM) / Scientific Linux 5.5 with*

*Alces HPC Software Stack*


Copyright © 2011 Alces Software Ltd All Rights Reserved

# TABLE OF CONTENTS


0.1. LICENSING INFORMATION....3
1. INTRODUCTION....4
1.1. SYSTEM INFORMATION....4
1.2. PROTECTING YOUR DATA AND INFORMATION....4
1.3. USING THIS GUIDE....4
2. CLUSTER ARCHITECTURE....5
2.1. LINUX OPERATING SYSTEM....5
2.2. BEOWULF CLUSTER ARCHITECTURE....5
2.3. CLUSTER SERVERS....6
2.4. SHARED DATA STORAGE CONFIGURATION....7
3. ACCESSING THE COMPUTE CLUSTER....8
3.1. LOGGING IN TO THE CLUSTER VIA THE COMMAND LINE....8
3.2. GRAPHICAL LOGIN ENVIRONMENT....9
a) USING REMOTE DESKTOP CONNECTIONS....9
b) NX CLIENT CONNECTIONS....9
3.3. CHANGING YOUR PASSWORD....12
a) CHANGING PASSWORDS WITH NIS....12
b) CHANGING PASSWORDS WITH LDAP....12
4. CONFIGURING YOUR USER ENVIRONMENT....13
4.1. LINUX SHELL CUSTOMISATION....13
4.2. MODULES ENVIRONMENT SWITCHER....13
a) DYNAMIC MODULE LOADING....13
b) SETTING UP YOUR DEFAULT ENVIRONMENT....15
c) LOADING MODULES FROM SCHEDULER JOBSCRIPTS....15
d) MANUALLY CONFIGURING YOUR USER ENVIRONMENT....16
5. USING COMPILERS AND LIBRARIES....17
5.1. AVAILABLE COMPILERS FOR SERIAL/BATCH JOBS....17
5.2. AVAILABLE COMPILERS FOR PARALLEL JOBS....17
5.3. ACCELERATED HPC LIBRARIES....18
5.4. COMPILER ENVIRONMENT VARIABLES FOR LIBRARIES....19
6. MESSAGE PASSING ENVIRONMENTS....20
6.1. MESSAGE PASSING INTRODUCTION....20
6.2. MPI INTERCONNECTS....20
a) ETHERNET NETWORKS....20
b) INFINIBAND FABRICS....21
6.3. SELECTING AN MPI IMPLEMENTATION....21
6.4. EXECUTING A PARALLEL APPLICATION....22
7. CLUSTER SCHEDULERS....23
7.1. CLUSTER SCHEDULER INTRODUCTION....23
7.2. TYPES OF JOB....23
8. GRID-ENGINE CLUSTER SCHEDULER....25




Copyright © 2011 Alces Software Ltd All Rights Reserved

8.1. USING GRID-ENGINE 6 CLUSTER SCHEDULER....25
8.2. JOB SCRIPTS AND QSUB DIRECTIVES....25
8.3. GRID-ENGINE PSEUDO ENVIRONMENT VARIABLES....28
8.4. JOB OUTPUT FILES....28
8.5. SUBMITTING NON-INTERACTIVE JOBS VIA QSUB....28
8.6. VIEWING THE STATUS OF A SUBMITTED JOB....29
8.7. SUBMITTING A PARALLEL JOB....30
8.8. SUBMITTING INTERACTIVE JOBS:....31
8.9. SUBMITTING AN ARRAY OF JOBS:....32
8.10. JOB DEPENDENCIES:....33
8.11. DELETING A SUBMITTED JOB....33
9. TORQUE / OPENPBS CLUSTER SCHEDULER....34
9.1. OVERVIEW OF TORQUE / OPENPBS....34
9.2. JOB SCRIPTS AND TORQUE DIRECTIVES....34
9.3. TORQUE PSEUDO ENVIRONMENT VARIABLES....36
9.4. JOB OUTPUT FILES....37
9.5. SUBMITTING NON-INTERACTIVE JOBS VIA QSUB....37
9.6. VIEWING THE STATUS OF A SUBMITTED JOB....38
9.7. SUBMITTING A PARALLEL JOB....38
9.8. EXECUTING AN INTERACTIVE JOB....39
9.9. SUBMITTING AN ARRAY OF JOBS:....40
9.10. REQUESTING EXCLUSIVE USE OF A COMPUTE NODE....41
9.11. DELETING A SUBMITTED JOB....41
10. LUSTRE PARALLEL FILESYSTEM....42
10.1. LUSTRE BACKGROUND....42
10.2. QUERYING FILESYSTEM SPACE....42
APPENDIX A: EXAMPLE PARALLEL JOB EXECUTION....44
APPENDIX B: EXAMPLE OPENMP JOB COMPILATION....46

## *0.1. LICENSING INFORMATION*

This work is © Copyright 2007-2011 Alces Software Ltd, All Rights Reserved.
Unauthorized re-distribution is prohibited.

The Alces Software HPC Cluster Toolkit is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. These packages are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details (http://www.gnu.org/licenses/).

A copy of the GNU Affero General Public License is distributed along with this product.
For more information on Alces Software, please visit: http://www.alces-software.com/.

# 1. INTRODUCTION

This quick reference guide is intended to provide a base reference point for users of an Alces Software configured RedHat, Centos or Scientific Linux 5 cluster installation. It shows basic examples of compiling and running compute jobs across the cluster and provides information about where you can go for more detail.

The primary audience for this guide is an HPC cluster user. All software is configured to be as close to the requirements of the original community supported distribution as is possible; more detailed documentation on specific packages are simple to find by following the links provided in this document.

## *1.1. SYSTEM INFORMATION*

In order to access the cluster, your system administrator will provide you with the following:

- The IP address and network information for the cluster
- A username and password to allow you to log in to the cluster

The software installed on your server nodes is designed to accept and run jobs from particular systems - these systems are called the ***login nodes***. In a cluster configuration, this may be the headnode system(s), or dedicated login nodes may be provided. It is not normally necessary to log-in directly to compute or worker nodes and your system administrator may prohibit direct access to non-interactive systems as part of your site information security policy.

## *1.2. PROTECTING YOUR DATA AND INFORMATION*

The software installed on your HPC cluster is designed to assist multiple users to efficiency share a large pool of compute servers, ensuring that resources are fairly available to everyone within certain configured parameters. Security between users and groups is strictly maintained, providing you with mechanisms to control which data is shared with collaborators or kept private.

When using the HPC cluster, you have a responsibility to adhere to the information security policy for your site which outlines acceptable behaviour and provides guidelines designed to permit maximum flexibility for users while maintaining a good level of service for all. Your system administrator can advise you of any specific rules covering the HPC system, but users are encouraged to:

- Change your system password as regularly as possible
- Only access machines that you have a valid requirement to use
- Adhere to your site information security policy
- Make backup copies of important or valuable data
- Remove temporary data after use

## *1.3. USING THIS GUIDE*

This quick-start user manual is intended to provide users with basic information about their HPC cluster. Your local cluster may be configured differently by your system administrator to suit the available hardware - if in doubt, contact your system administrator for further assistance.


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 2. CLUSTER ARCHITECTURE

## 2.1. LINUX OPERATING SYSTEM

Your HPC cluster is installed with a 64-bit Linux operating system. CentOS and Scientific Linux (SL) are freely available Linux distributions based on RedHat Enterprise Linux. Both projects attempt to maintain 100% compatibility with the corresponding Enterprise distribution by recompiling packages and updates from source as they are released by RedHat. These open-source distributions are supported by a thriving community of users and developers with many support sites, wikis and forums to assist adopters with its configuration.

Redhat distribution version numbers have two parts - a major number corresponding to the Enterprise Linux version and minor number which corresponds to the Enterprise update set. Redhat 4.6 therefore corresponds to RedHat Enterprise Linux 4 update 6. Since 64-bit extensions were first integrated into X86 compatible processors in 2003, 64-bit capable operating systems have allowed users to take advantage of both 32-bit and 64-bit computing. Each system in your Linux cluster is installed with either Redhat, CentOS or Scientific Linux 5 64-bit edition.

## 2.2. BEOWULF CLUSTER ARCHITECTURE

The purpose of a *Beowulf* or *loosely-coupled* Linux cluster is to provide a large pool of compute resource which can be easily managed from a single administration point. Typically, worker nodes are identically configured with other special-purpose servers to provide cluster services including storage, management and remote user access. Each node in a Linux cluster is installed with an operating system - often, *personality-less* system configurations are adopted on compute nodes to help reduce administration overhead. Typically, compute nodes have a minimal operating system installation to minimize unwanted services and help reduce jitter across the cluster. The configuration of the node operating system is controlled by the cluster provisioning software.

The operating systems on special purpose nodes are configured specifically for their purpose in the cluster. Machines are categorized by the roles they perform, for example:

- *Head node system*; provides cluster management and scheduler services
- *Interactive system*; provides software development environment
- *Compute node system*; used solely for executing compute jobs
- *Storage system*; provides data services to other nodes in the cluster
- *Gateway system*; provide a secure pathway into the cluster

Depending on your cluster size and configuration, some physical servers may perform several different roles simultaneously; for example, the headnode of a small cluster may provide cluster management, scheduler servers, an interactive development environment and storage services to the compute nodes in your cluster. All system administration is usually performed from the cluster headnode or administration machine;


Copyright © 2011 Alces Software Ltd All Rights Reserved

the goal of the installed cluster management utilities is to reduce the requirement to log in to each compute node in turn when configuring and managing the system.

Non-administrative cluster users typically access the cluster by submitting jobs to the scheduler system; this may be performed remotely (by using a scheduler submission client installed on their local workstation) or by submitting jobs from the cluster login or interactive nodes. It is not normally necessary for users to log in directly to the compute nodes - many cluster administrators disallow this to encourage the fair sharing of resources via the cluster scheduler system.

Your cluster may be configured with multiple different networks dedicated to different purposes. These may include:

- Interactive user data and job scheduler information
- Operating system provisioning
- Lights-out-management, IPMI or SNMP traffic
- Storage networking
- Message-passing interconnect traffic
- Graphics rendering / visualization interconnect

Depending on the size of the cluster, some of these functions may be shared on a single network, or multiple networks may be provided for the same function. When using your cluster for the first time, you should consider which network to use for message-passing functions, as these operations are sensitive to message latency and network bandwidth.

## *2.3. CLUSTER SERVERS*

Your cluster is configured with a number of different server systems that perform different functions in the cluster. Typical systems may include:

- Cluster login nodes; e.g.
  - **`login1.university.ac.uk, login2.university.ac.uk`**
- Compute node servers; e.g.
  - **`comp00-comp23`** - standard compute nodes
  - **`bigmem00-bigmem03`** - high memory compute nodes
- Cluster service nodes; e.g.
  - **`headnode1`** - primary headnode system
  - **`headnode2`** - secondary headnode system
  - **`mds1`** - Lustre filesystem primary metadata server
  - **`nfs1`** - Network file system primary server
  - **`oss1, oss2`** - Lustre object storage servers

As a cluster user, you will mostly be running jobs on *compute node servers* via the **cluster job scheduler**


Copyright © 2011 Alces Software Ltd All Rights Reserved

(often also called ***cluster load-balancing software*** or the ***batch system***). Each compute node is equipped with high performance multi-core processors, tuned memory systems and often have a small amount of local temporary scratch space assigned on their hard disks. In contrast, cluster service nodes often feature specialised hardware and slower, low-power processors to enable them to perform their intended purpose efficiently. Your compute jobs will execute fastest on compute nodes and should not be started on cluster service nodes unless directed by your local administrator.

Each compute node is installed with a minimal operating system that allows it to boot and function as a Linux host to run compute jobs. All application software, compilers, libraries and user data is held centrally on your cluster service nodes with very few software packages installed on individual compute nodes. This helps to ensure that all nodes are identically configured, allowing jobs submitted to the cluster scheduler to run equally well whichever node they are executed on. The operating system and system software installed on compute nodes is controlled by your system administrator from a central location; individual nodes may be automatically re-installed between successive job executions. Users should only store temporary data on compute node hard disks as scratch areas may be cleaned between job executions.

## 2.4. SHARED DATA STORAGE CONFIGURATION

Your cluster system may be configured with a number of different file storage areas mounted on different servers for administrative and data sharing purposes. These areas are designed to provide user data, system libraries and configuration information to the nodes from a single, central location. By using shared storage, the management overhead of a large cluster can be significantly reduced as there is no need to copy data to every node in the cluster for running jobs. By default, the following shared storage areas are typically configured on the cluster:

| *Storage mount point on nodes* | *File server* | *Purpose* |
|---|---|---|
| **/users** | headnode or storage servers | Shared storage area for users |
| **/opt/gridware** | headnode or storage servers | Optional software including cluster scheduler |
| **/opt/alces** | headnode or configuration server | Cluster configuration information |
| **/scratch or /tmp** | Local compute node disk or storage servers | High-speed transient data storage |

Where data storage areas are shared via NFS from the central fileserver, compute nodes are configured with both a unique scratch storage area (mounted under **/tmp** or **/scratch**) and a shared storage area (mounted under **/users**). Any applications or services installed on compute nodes that use the **/tmp** area to store temporary data can make use of individual compute node disks without affecting the performance or capacity of the centrally stored data. Users should be aware that data stored on node hard disks may be removed after their job has finished executing - always copy data to be retained to your home-directory as part of your jobscript.


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 3. ACCESSING THE COMPUTE CLUSTER

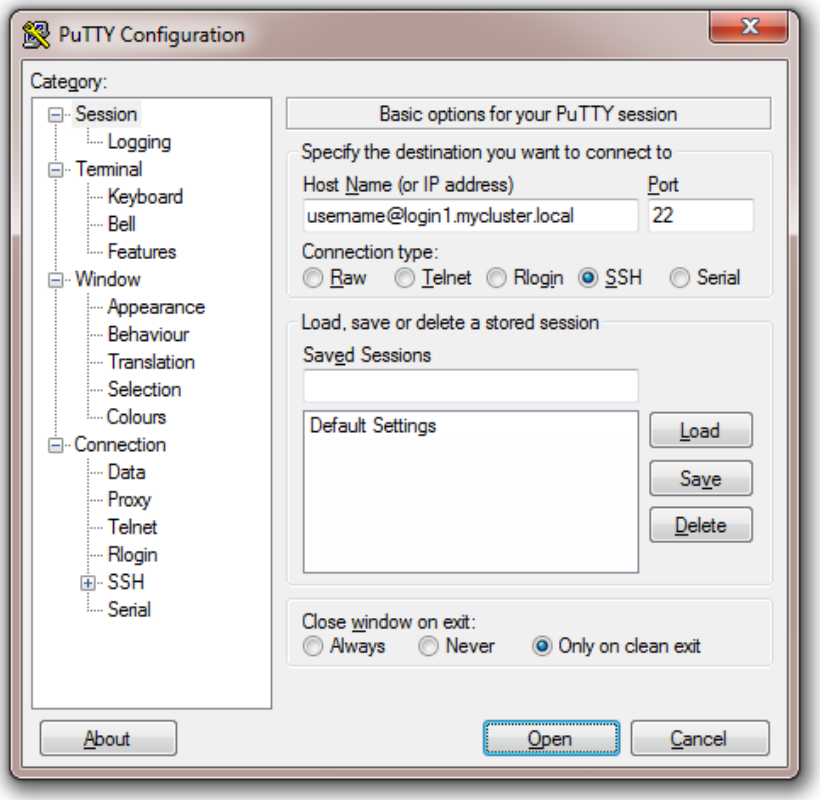
Cluster users typically work from the *login nodes* of an HPC cluster, using them to submit jobs, compile code and monitor the status of running jobs. Larger clusters may have one or more dedicated login nodes, whereas smaller machines may have just one login node hosted by the headnode system. Your system administrator will provide you with the host address of the cluster login node(s) as well as your user name and password to access the cluster.

## *3.1. LOGGING IN TO THE CLUSTER VIA THE COMMAND LINE*

Users can log-in to a cluster login node using secure-shell (SSH) from your desktop system or another machine on the network. To log in from a UNIX, Linux or MacOS X system, use the *ssh* command:

```
[username@workstation ~]$ ssh username@login1.mycluster.local
Password: ********

[username@login1 ~$]
```

To log in from a Microsoft Windows workstation, users must download and run an SSH client application; the open-source *PuTTY* package is a popular choice, as shown below:





Copyright © 2011 Alces Software Ltd All Rights Reserved

## 3.2. GRAPHICAL LOGIN ENVIRONMENT

If your cluster login node(s) have been configured with graphical login service, authorized users can obtain an interactive desktop session from a network connected workstation or laptop. Your system administrator will let you know which services are configured and if your user account is enabled for graphical access.

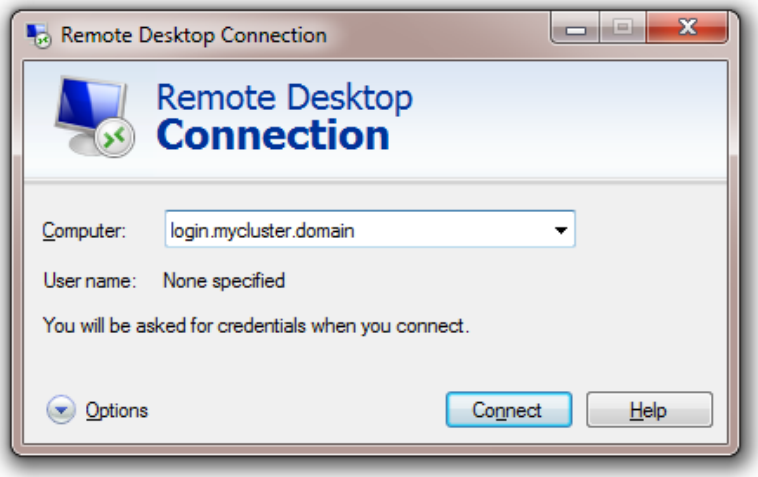
### a) USING REMOTE DESKTOP CONNECTIONS

When configured, the *XRDP* service provides a quick and easy method to access your cluster system and run graphical applications using a *Remote Desktop* client. You can connect to the cluster headnode from any Linux, Solaris, Mac OSX or Microsoft Windows client with a suitable client application installed. By default, the *XRDP* application runs on port 3389 - the default for Windows Terminal Services or Remote Desktop clients.

For Linux or MacOS X clients, use the *rdesktop* client to connect to the headnode machine:

```
[username@workstation ~]# rdesktop headnode:3389
```

For Microsoft Windows clients, start the *Remote Desktop Connection* application, enter the hostname or IP address of your cluster login node system and press the *connect* button:




### b) NX CLIENT CONNECTIONS

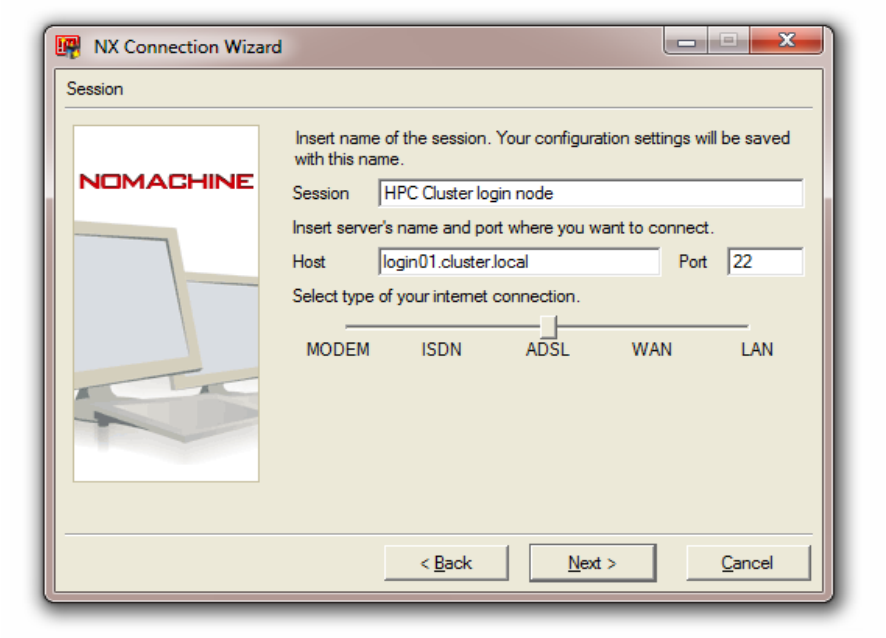
The NX series of display libraries provides a convenient, high-performance method of running a graphical desktop from the cluster master or login node on your local client system. Open-source client binaries are provided by the *NoMachineNX* project for the majority of client desktops including Microsoft Windows, Mac OSX, Solaris and Linux. Visit the following URL to download a client package suitable for your client system:

http://www.nomachine.com/download.php

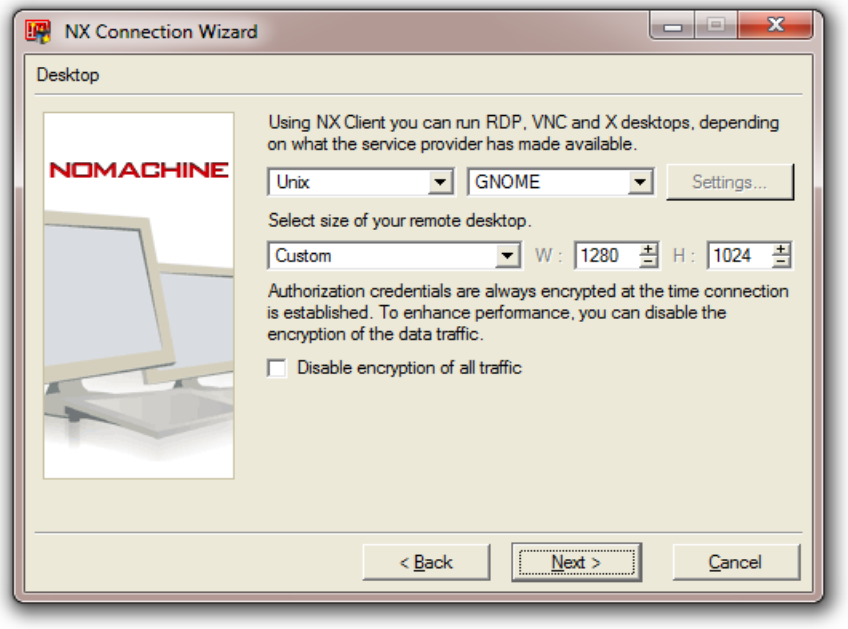
After installation, start the ***NX Connection Wizard*** to configure the settings for your HPC cluster; enter a recognisable name to describe the connection and the hostname or IP address details of your cluster login


Copyright © 2011 Alces Software Ltd All Rights Reserved

or master node machine. Set the slider to match the network connection speed between your client machine and the HPC cluster:




Click the ***Next*** button and select a "UNIX GNOME" desktop - the standard for Linux RedHat, CentOS and Scientific Linux systems. Configure the size of your desktop by selecting one of the pre-defined options, or select ***custom*** and enter the desired width and height in pixels:


NX Connection Wizard
Desktop
NOMACHINE
Using NX Client you can run RDP, VNC and X desktops, depending on what the service provider has made available.
Unix
GNOME
Settings...
Select size of your remote desktop.
Custom
W :
1280
H :
1024
Authorization credentials are always encrypted at the time connection is established. To enhance performance, you can disable the encryption of the data traffic.
Disable encryption of all traffic
< Back
Next >
Cancel

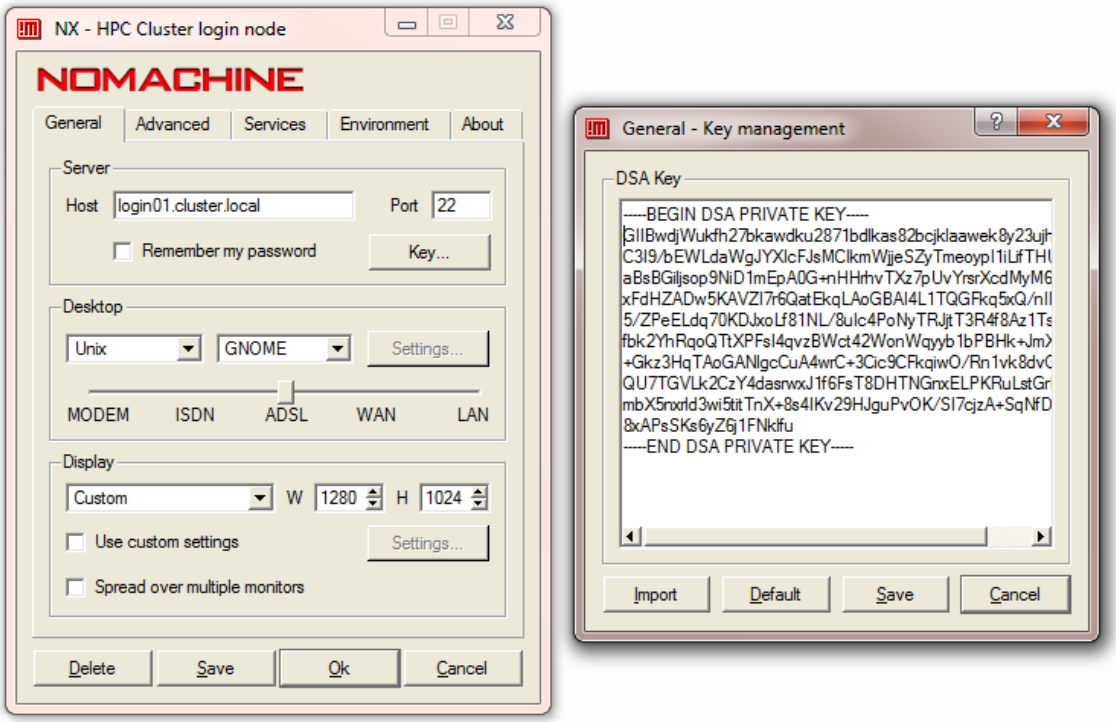
Click the ***Next*** button and click on the "Show Advanced Settings" check-box - click ***Next*** again.

The advanced settings dialogue box allows users to enter the connection key used to connect to the NX server running on your HPC cluster. This key is specific to your cluster and ensures that only clients that

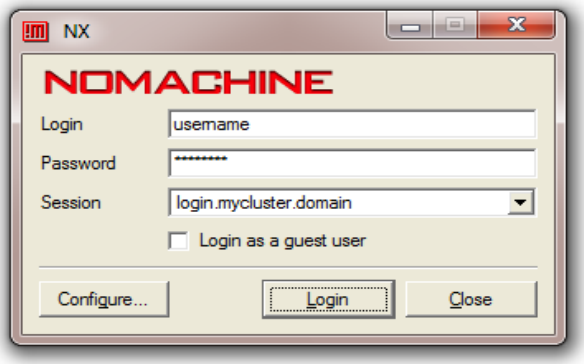

Copyright © 2011 Alces Software Ltd All Rights Reserved

identify themselves with the correct key are allowed to connect. Your system administrator will provide you with a copy of the keyfile, stored on your HPC cluster login or master node(s).

Click on the ***Key*** button on the configuration dialogue and enter the key provided by your system administrator. Press the ***Save*** button on the key entry box, then press ***Save*** again on the main configuration box to save your changes.




Enter your cluster username and password in the dialogue boxes provided and press the ***Login*** button. Connection messages will be displayed while your client negotiates with the NX server before your remote desktop session is displayed.




Contact your system administrator for assistance if your remote desktop session is not started as expected; they may ask for the diagnostic messages displayed during negotiation to assist in troubleshooting the connection problem.


Copyright © 2011 Alces Software Ltd All Rights Reserved

## 3.3. CHANGING YOUR PASSWORD

Your system administrator will have provided you with a login password or SSH-key access to the compute cluster, allowing you to login and use the attached resources. We recommend that you change your system password as often as possible and adhere to the recommendations explained in your site information security policy.

### a) CHANGING PASSWORDS WITH NIS

If your cluster uses Network Information Services (NIS), you can change your user password using the **yppasswd** command on the command line when logged in to the cluster.

```
[user@login01 ~]$ yppasswd
Changing NIS account information for user on headnode1.cluster.local.

Please enter old password: ********
Changing NIS password for user on headnode1.cluster.local.

Please enter new password: ********
Please retype new password: ********

The NIS password has been changed on headnode1.cluster.local.
```

Your user password can be reset by your system administrator if you have forgotten what it is - a temporary password may be issued to you to allow you to log in; use the **yppasswd** command to change the temporary password to one that you can remember.

### b) CHANGING PASSWORDS WITH LDAP

For clusters that are configured to use site-wide LDAP services, access to login machines will be authorized using your standard LDAP password used on other site machines. Contact your system administrator for assistance changing your LDAP password for your site.


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 4. CONFIGURING YOUR USER ENVIRONMENT

## 4.1. LINUX SHELL CUSTOMISATION

As a cluster user, you may need access to specific applications, compilers, libraries and utilities in order to run your compute jobs on the machine. When logging in, the default command-line access method is via the bash shell interpreter. Bash can be configured to provide convenient access to the different software you use need to help simplify usage of the cluster.

In your home directory, a file called **.bashrc** contains a list of customisations which are executed by the bash shell every time you log in. As your home directory is shared across all machines in the compute cluster, the **.bashrc** file is also executed automatically when your compute jobs are executed by cluster compute nodes.

The **.bashrc** file may be customised by users by simply editing it in a text editor (e.g. **nano, emacs** or **vim**). After changing the file, users must logout and login again, causing the bash shell to re-read the file and apply any changes made.

When running applications or developing and compiling code to run on an HPC cluster, there are a number of different compiler and library options which users may need to build or link against. Many different packages have similar names for commands and it can quickly become difficult to instruct the system exactly which tools and utilities should be used. Linux shell interpreters (e.g. bash, csh, tsh) use a series of **environment variables** to provide convenient short-cuts for running binaries and locating system libraries. Common variables include:

| Environment variable | Usage |
|---|---|
| **$PATH** | Stores the search path for executable tools and utilities |
| **$LD_LIBRARY_PATH** | Stores the search path for library files that may be used by executed applications |
| **$MANPATH** | Stores the search path for user manual pages describing different utilities and tools |
| **$USER** | Stores your current user name |
| **$PWD** | Stores your current working directory |

## 4.2. MODULES ENVIRONMENT SWITCHER

The *module* environment switcher provides a convenient method of setting your environment variables for the different libraries installed on your cluster. When invoked, the utility modifies the running session, setting the **PATH, LD_LIBRARY_PATH** and **MAN_PATH** variables as requested.

### a) DYNAMIC MODULE LOADING

Modules are installed in a shared area (e.g. **/opt/gridware/modules**) to make them available across the


Copyright © 2011 Alces Software Ltd All Rights Reserved

cluster on all login and compute nodes. All users logged into any cluster node can use module commands with identical syntax.

To view the different module environments available on your cluster, use the "**module avail**" command:

```
[user@login01 /]# module avail

---------------------/usr/share/Modules/modulefiles-----------------------
dot                      mpi/openmpi-1.2.6_gcc    null
module-cvs               mpi/openmpi-1.2.6_intel  switcher/1.0.13(default)
module-info              mpi/openmpi-1.2.6_pgi
modules                  mpi/openmpi-1.3.2_gcc

------------------------- /opt/gridware/modules---------------------------
apps/gcc/beast        apps/gcc/emboss       apps/gcc/pft3dr
apps/gcc/blast        apps/gcc/hmmer        apps/gcc/spider
apps/gcc/bsoft        apps/gcc/imod         apps/gcc/velvet
apps/gcc/bsoft32      apps/gcc/mafft        libs/perl-cpan
apps/gcc/eman         apps/gcc/mrbayes      mpi/gcc/openmpi/1.4.1
apps/gcc/eman2        apps/gcc/paml44

[user@login01 /]#
```

- Use the "**module load**" command to enable a new environment from the list displayed. The **load** command may be used multiple times to include settings from multiple different profiles.
- Use the "**module list**" command to list currently loaded modules.
- Use the "**module unload**" command to unload an already loaded module.
- Use the "**module purge**" command to clear all loaded modules.

The **"module display"** and **"module help"** commands can help users to determine which environment variables are managed by different modules:

```
[user@login01 /]#  module help libs/gcc/gsl/1.14

----------- Module Specific Help for 'libs/gcc/gsl/1.14' ----------

Adds `gsl-1.14' to your environment variables

###############
ENV after load
###############
GSLDIR -> Base path of library
GSLLIB -> lib path
GSLBIN -> bin path
GSLINCLUDE -> include path
Adds GSLLIB to LD_LIBARY_PATH
Adds GSLBIN to PATH
Adds gsl ManPages to MANPATH
Adds GSLINCLUDE to C_INCLUDE_PATH and CPLUS_INCLUDE_PATH
Adds necessary flags to build/link against the library to CFLAGS and
LDFLAGS

[user@login01 /]#
```

After loading an application module, the base location of the application is set in an environment variable to provide quick access to any application help or example files bundled with the application. For example:


Copyright © 2011 Alces Software Ltd All Rights Reserved

```
[user@login01 /]# module load apps/mrbayes
[user@login01 /]# echo $mrbayes
/opt/gridware/apps/mrbayes/1.3.9/

[user@login01 /]# cd $mrbayes
[user@login01 1.3.9]# ls
AUTHORS      COPYING    doc   examples  MANUAL    scripts
TUTORIAL     bin        lib   lib64     VERSION
[user@login01 1.3.9]#
```

### b) SETTING UP YOUR DEFAULT ENVIRONMENT

Users can request that certain modules are loaded every time they log into the cluster by using the **module initadd** and **module initrm** commands. These commands manipulate a file held in the users' home directory called '**.modules**', causing the system to pre-load the desired modules as the user logs in. This is particularly useful for users who run a fixed set of applications and require the same environment to be configured each time they log into the cluster.

- Use the "**module initadd**" command to cause a module to be loaded on login.
- Use the "**module initprepend**" command to cause a module to be loaded on login, before any other modules that are already being loaded on login.
- Use the "**module initlist**" command to list the modules that will be loaded on login.
- Use the "**module initrm**" command to remove a module listed to load on login.
- Use the "**module initclear**" command to clear all modules from being loaded on login.

Your system administrator can also configure the system to automatically load one or more default modules for users logging in to the cluster. Use the **module list** command to determine which modules are loaded when you log in.

### c) LOADING MODULES FROM SCHEDULER JOBSCRIPTS

When submitting jobs through the scheduler system, users may wish to use **module** commands to configure their environment variable settings before running jobs. Three methods can be used to use environment variables in job submission scripts:

- Permanently add the required modules to be loaded for your login environment
- Load modules before submitting jobs and use the -V scheduler directive to export these variables to the running job
- Use a "**module load**" command in your jobscript to load additional modules

Users incorporating a "**module load**" command in their job script should remember to source the modules profile environment (e.g. **/etc/profile.d/modules.sh**) in their jobscripts before loading modules to allow the environment to be properly configured; e.g.


Copyright © 2011 Alces Software Ltd All Rights Reserved

```
#!/bin/bash
# This is an example job script
# Scheduler directives are shown below
#$ -j y -o ~/myoutputfile
# Configure modules
. /etc/profile.d/modules.sh

# Load modules
module load mpi/gcc/openmpi/1.5.2

echo "Job starting at `date`"
~/test/myapplication -i ~/data/intputfile4
```

### d) MANUALLY CONFIGURING YOUR USER ENVIRONMENT

The *modules* environment switcher is installed for the convenience of users to assist in managing multiple different compilers, libraries and software development environments. If you prefer, your user environment can be set-up manually, bypassing the modules system for ultimate flexibility when configuring how you use the cluster system. This can be achieved by manually setting your environment variables, specifying the individual libraries to build against when compiling code or storing your desired libraries in your home directories and addressing them directly. Although these methods are quite valid for advanced users, please be aware that bypassing the global modules configuration may reduce the ability of your system administrator to assist you if your applications do not later execute as expected.


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 5. USING COMPILERS AND LIBRARIES

## 5.1. *AVAILABLE COMPILERS FOR SERIAL/BATCH JOBS*

As well as pre-compiled binary applications, many HPC users also write their own software to solve complex mathematical and scientific problems. Users are encouraged to compile new applications on the login node systems, testing them for correct behaviour before submitting them as jobs to run across the compute cluster. Your compute cluster may be configured with a dedicated pool of resources for development and testing which is distinct from the main cluster.

A number of different open-source and commercial Linux compilers are available, providing different optimisations, tuning options and feature sets. The default open-source GNU compiler suite is provided as part of your Linux distribution and is available by default on all compute clusters. The following table summarises the compiler commands available on HPC clusters:

| Package | License type | Language | | | | |
|---|---|---|---|---|---|---|
| | | C | C++ | Fortran77 | Fortran90 | Fortran95 |
| GNU | Open-source | `gcc` | `g++` | `gfortran` | `gfortran` | `gfortran` |
| Open64 | Open-source | `opencc` | `openCC` | `n/a` | `openf90` | `openf95` |
| Intel | Commercial | `icc` | `icc` | `ifort` | `ifort` | `ifort` |
| Portland Group | Commercial | `pgcc` | `pgCC` | `pgf77` | `pgf90` | `pgf95` |
| Pathscale | Commercial | `pathcc` | `pathCC` | `n/a` | `pathf90` | `pathf95` |

Commercial compiler software requires a valid license for your site before users can compile and run applications. If your cluster is properly licensed, environment modules are provided to enable you to use the compiler commands listed above. If your cluster is not licensed for commercial compilers, or you have not loaded a compiler environment module, the open-source **GNU** compilers appropriate to your Linux distribution will be available for you to use.

## *5.2. AVAILABLE COMPILERS FOR PARALLEL JOBS*

When compiling source code for parallel jobs, users may prefer to use a compiler which is capable of creating binaries compatible with a message-passing interface (MPI). The following compiler commands are automatically available after loading an MPI environment module (e.g. OpenMPI, MvAPICH):

| Language | C | C++ | Fortran77 | Fortran90 | Fortran95 |
|---|---|---|---|---|---|
| Command | **`mpicc`** | **`mpiCC`** | **`mpif77`** | **`mpif77`** | **`mpiCC`** |


Copyright © 2011 Alces Software Ltd All Rights Reserved

## 5.3. ACCELERATED HPC LIBRARIES

To assist developers of high performance applications and utilities, a number of optimised libraries are available to provide commonly used functions including Fourier transforms, linear algebra, sparse solvers, vector mathematics and other routines. Often these libraries are tuned to take advantage of the latest CPU instruction sets to accelerate their performance on modern computer systems - a well tuned library set can improve the performance of applications run across the cluster by up to 30%.

Your HPC cluster has been installed with a set of high performance libraries that are appropriate for the hardware and software components that make up the system. Each library is installed in a shared location available to all compute nodes - users may add libraries into their environment by loading the module for the library set they wish to use. See section 4.2 for more information about loading environment modules.

The following library packages are commonly available on HPC compute clusters:

| Library name | License type | Library type | Library location variable |
|---|---|---|---|
| ATLAS | Open-source | Auto-tuned linear algebra software | `$ATLASLIB` |
| BLAS | Open-source | Basic linear algebra sub-programs (F77 implementation) | `$BLASLIB` |
| CBLAS | Open-source | Basic linear algebra sub-programs (C implementation) | `$CBLASLIB` |
| FFTW2-double | Open-source | Discrete Fourier transform (double-precision version 2) | `$FFTWDIR` |
| FFTW2-float | Open-source | Discrete Fourier transform (single-precision version 2) | `$FFTWDIR` |
| FFTW3 | Open-source | Discrete Fourier transform (latest stable version 3 for single/float, double and long-double precision) | `$FFTWDIR` |
| LAPACK | Open-source | Linear algebra package | `$LAPACKDIR` |
| Intel MKL | Commercial | Intel kernel math library suite | `$MKL` |
| ACML | Commercial | AMD core math library suite | `$ACML` |

Commercial libraries require a valid license for your site before users can compile and run applications using them. If your cluster is properly licensed, environment modules are provided to enable you to use the libraries listed above. If your cluster is not licensed for commercial libraries, or you have not loaded a compiler environment module, the standard system libraries appropriate to your Linux distribution will be available for you to use.

Some high-performance libraries require pre-compilation or tuning before they can be linked into your applications by the compiler you have chosen. When several different compilers are available on your cluster, libraries may have been prepared using multiple different compilers to provide greater flexibility for users who include them from their source code. The *modules* environment may list some high-performance libraries multiple times, one for each compiler they have been prepared on. Users should load the library environment module that matches the compiler they intend to use to compile their applications.


Copyright © 2011 Alces Software Ltd All Rights Reserved

## 5.4. COMPILER ENVIRONMENT VARIABLES FOR LIBRARIES

When a module containing accelerated libraries is loaded, the system automatically updates relevant system environment variables to assist users in writing portable code and makefiles which can be used with multiple different libraries. The **$CPPFLAGS** and **$LDFLAGS** variables are dynamically updated to include the relevant include directives for compilers as library modules are loaded; e.g.

```
[user@login01 /]# module load libs/gcc/fftw3/3.2.2
[user@login01 /]# echo $CPPFLAGS
-I/opt/gridware/libs/gcc/fftw3/3_2_2/include
[user@login01 /]# echo $LDFLAGS
-L/opt/gridware/libs/gcc/fftw3/3_2_2/lib/
[user@login01 /]#
```


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 6. MESSAGE PASSING ENVIRONMENTS

## 6.1. MESSAGE PASSING INTRODUCTION

A message-passing interface (MPI) is an API specification that allows processes to communicate with one another by sending and receiving messages. Typically used for parallel programs, an MPI can allow processes on one node to communicate with processes running on an entirely separate node, providing greater flexibility and large scale applications to be executed across an HPC cluster. Communication messages between nodes can be transported via a number of common interconnect networks including Ethernet, Myrinet or Infiniband.

A number of different commercial and open-source MPI implementations are available for C, C++ and Fortran code and there may be multiple different options installed on your compute cluster. Applications written to use an MPI are often compiled to include the MPI libraries they require, using the compilers and (optionally) high-performance libraries described in section 5.

## 6.2. MPI INTERCONNECTS

The network transport used for MPI messages will depend on the hardware provided with your HPC cluster. Most basic clusters have a gigabit Ethernet interconnect which is shared between a number of different tasks. More complex clusters may have one or more dedicated gigabit Ethernet networks for MPI traffic. Clusters that process large amounts of parallel workload may be installed with a high-bandwidth, low-latency interconnect such as Infiniband, Myrinet or 10-gigabit Ethernet. These specialist networks are especially designed to provide the fastest message passing systems available at bandwidths of multiple gigabytes per second. Your system administrator will provide you with information on the different MPI networks installed on your cluster, and can offer advice on how to get the best performance for your applications.

### a) ETHERNET NETWORKS

The majority of HPC clusters incorporate one or more gigabit Ethernet networks into their design. Popular for their low cost of ownership, relatively high performance of 1Gbps per link and flexible CAT5E cabling, gigabit Ethernet networks are now commonly being deployed across large campuses and to users' desktops. Your cluster may have a dedicated MPI gigabit Ethernet network, or might share a network with other tasks. 10-gigabit Ethernet links are also becoming increasingly popular and are often used simply as a higher bandwidth version of gigabit Ethernet. Suitable MPI implementations for Ethernet networks include:

- OpenMPI; developed by merging the popular FT-MPI, LA-MPI and LAM/MPI implementations
- MPICH; an implementation of the MPI-1.1 standard
- MPICH2; an implementation of the MPI-2.0 standard
- Intel MPI; a commercial MPI implementation available for both X86 and IA64 architectures


Copyright © 2011 Alces Software Ltd All Rights Reserved

b) INFINIBAND FABRICS

Available as add-on interfaces to most compute node system, Infiniband host-channel adapters allow compute nodes to communicate with others at speeds of up to 32Gbps and latencies of around 2 microseconds. Infiniband also provides RDMA (remote direct memory access) capabilities to help reduce the CPU overhead on compute nodes during communication. Infiniband can transmit and receive messages in a number of different formats including those made up of Infiniband *verbs*, or *TCP* messages similar to Ethernet networks. Suitable MPI implementations for Infiniband include:

- OpenMPI; developed by merging the popular FT-MPI, LA-MPI and LAM/MPI implementations
- MVAPICH; an implementation of the MPI-1.1 standard
- MVAPICH2; an implementation of the MPI-2.0 standard
- Intel MPI; a commercial MPI implementation available for both X86 and IA64 architectures

## 6.3. SELECTING AN MPI IMPLEMENTATION

Each MPI available on your cluster has been pre-built for the available hardware and different compiler options available on your site. Each option is configured by default to automatically run over the best performing interconnect network available. Contact your system administrator if you need assistance running a parallel job over a different interconnect fabric.

Use the "**`module load`**" command to load the appropriate environment module for the MPI required:

```
[user@login01 /]$ module avail

---------------------- /opt/gridware/modulefiles/ -------------------------
compilers/intel/11_1_072/32         libs/intel/fftw3/3_2_2/fftw3
compilers/intel/11_1_072/64         mpi/gcc/mpich2/1_3/mpich2
libs/gcc/atlas/3_9_32/atlas         mpi/gcc/mvapich/1_2rc1/mvapich
libs/gcc/blas/1                     mpi/gcc/mvapich2/1_5_1p1/mvapich2
libs/gcc/cblas/1/cblas              mpi/gcc/openmpi/1_5/openmpi
libs/gcc/fftw2/2_1_5/fftw-double    mpi/intel/mpich2/1_3/mpich2
libs/gcc/fftw2/2_1_5/fftw-float     mpi/intel/mvapich/1_2rc1/mvapich
libs/gcc/fftw3/3_2_2/fftw3          mpi/intel/mvapich2/1_5_1p1/mvapich2
libs/gcc/lapack/3_3_0/lapack        mpi/intel/openmpi/1_5/openmpi
libs/intel/fftw2/2_1_5/fftw-double  torque/2.5.3(default)
libs/intel/fftw2/2_1_5/fftw-float

[user@login01 /]$ module load mpi/gcc/mpich2
[user@login01 /]$
```

If users are developing source code that integrates multiple compiler, MPI and HPC libraries, they must remember to load all the modules required for the compilation, e.g.

```
[user@login01 /]$ module load compilers/intel libs/intel/fftw3 mpi/intel/mpich2
```


Copyright © 2011 Alces Software Ltd All Rights Reserved

Users may prefer to include commonly used environment modules into their default environment to ensure that they are loaded every time they login. See section 4.2 for more information about loading environment modules. Appendix A shows an example compilation of a parallel application and its execution using OpenMPI and the grid-engine cluster scheduler.

## 6.4. EXECUTING A PARALLEL APPLICATION

After compiling your parallel application, it can be submitted for execution via the cluster job scheduler system. Most job schedulers use jobscripts to instruct the scheduler how to run your application - see section 7 for more information on how to create jobscripts.

Your cluster scheduler automatically determines which compute nodes are available to run your parallel application, taking into account the topology of the requested interconnect network and any existing jobs running on the cluster. The cluster scheduler communicates this information to the selected MPI implementation via a **machinefile** which lists the hostnames of compute nodes to be used for the parallel execution. In order to control how many compute cores to use on each multi-CPU compute node, users must instruct the scheduler to allocate the correct number of cores to the job. Section 7 provides examples of the different options which can be used to pass this information to the cluster scheduler.


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 7. CLUSTER SCHEDULERS

## 7.1. CLUSTER SCHEDULER INTRODUCTION

Your HPC cluster is managed by a **cluster scheduler** - also known as the *batch scheduler*, *workload manager*, *queuing system* or *load-balancer*. This application allows multiple users to fairly share the managed compute nodes, allowing system administrators to control how resources are made available to different groups of users. A wide variety of different commercial and open-source schedulers are available for compute clusters, each providing different features for particular types of workload. All schedulers are designed to perform the following functions:

- Allow users to submit new jobs to the cluster
- Allow users to monitor the state of their queued and running jobs
- Allow users and system administrators to control running jobs
- Monitor the status of managed resources including system load, memory available, etc.

More advanced schedulers can be configured to implement policies that control how jobs are executed on the cluster, ensuring fair-sharing and optimal loading of the available resources. Most schedulers are extendible with a variety of plug-in options for monitoring different metrics, reporting system usage and allowing job submission via different interfaces. The scheduler system available on your compute cluster will depend on how your system administrator has configured the system - they will be able to advise you on how your HPC cluster is set up.

When a new job is submitted by a user, the cluster scheduler software assigns compute cores and memory to satisfy the job requirements. If suitable resources are not available to run the job, the scheduler adds the job to a queue until enough resources are available for the job to run. Your system administrator can configure the scheduler to control how jobs are selected from the queue and executed on cluster nodes. Once a job has finished running, the scheduler returns the resources used by the job to the pool of free resources, ready to run another user job.

## 7.2. TYPES OF JOB

Users can run a number of different types of job via the cluster scheduler, including:

- **Batch jobs**; non-interactive, single-threaded applications that run only on one compute core
- **Array jobs**; two or more similar batch jobs which are submitted together for convenience
- **SMP jobs;** non-interactive, multi-threaded applications that run on two or more compute cores on the same compute node
- **Parallel jobs;** non-interactive, multi-threaded applications making use of an MPI library to run on multiple cores spread over one or more compute nodes
- **Interactive jobs;** applications that users interact with via a command-line or graphical interface

Non-interactive jobs are submitted by users to the batch scheduler to be queued for execution when suitable resources are next available. Input and output data for non-interactive jobs are usually in the form of files read from and written to shared storage systems - the user does not need to remain logged into


Copyright © 2011 Alces Software Ltd All Rights Reserved

the cluster for their jobs to run. Scheduler systems provide a mechanism for collecting the information output by non-interactive jobs, making it available as files for users to query after the job has completed. Non-interactive jobs are usually submitted using a ***jobscript*** which is used to direct the scheduler how to run the application. The commands syntax used in the job script will depend on the type and version of scheduler installed on your HPC cluster.

Interactive jobs are commonly submitted by users who need to control their applications via a graphical or command-line interface. When submitted, the scheduler attempts to execute an interactive job immediately if suitable resources are available - if all nodes are busy, users may choose to wait for resources to become free or to cancel their request and try again later. As the input and output data for interactive jobs is dynamically controlled by users via the application interface, the scheduler system does not store output information on the shared filesystem unless specifically instructed by the application. Interactive jobs only continue to run while the user is logged into the cluster - they are terminated when a user ends the login session they were started from.


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 8. GRID-ENGINE CLUSTER SCHEDULER

## 8.1. USING GRID-ENGINE 6 CLUSTER SCHEDULER

Grid-Engine (GE) is a cluster scheduler designed to manage the resources available on your cluster machines and allow multiple users to securely and efficiently submit work to a shared resource. Grid-engine provides a simple mechanism for users to submit batch and parallel jobs from interactive nodes into centralized queues and have job results delivered to a designated location.

A typical grid-engine installation requires a *qmaster* server (normally the headnode or a cluster service node), one or more submit hosts from where users can submit jobs (typically a login or headnode server) and a number of execution hosts where jobs are run. The process for running a job through grid-engine is:

- Prepare the application or binary file to run on a cluster node
- Create a **jobscript** to run the application with the required parameters
- Select the grid-engine directives required to control how your job is run
- Submit the jobscript to GE from the cluster login or master node

The steps below indicate how to perform these steps to submit different types of job to the grid-engine scheduler.

## 8.2. JOB SCRIPTS AND QSUB DIRECTIVES

A **jobscript** usually takes the form of a simple shell script containing a list of tasks to perform in order to run your job. These may include simple commands (e.g. printing a status message, copying or moving files, etc.) or calling a binary application to execute with parameters. An example jobscript is shown below:

```
#!/bin/bash
# This is an example job script
# Grid-engine directives are shown below

# Export my current environment variables
#$ -V
# Merge stdout and stderr
#$ -j y
echo "Job starting at `date`"
~/test/myapplication -i ~/data/intputfile4
```

Lines preceded by a **'#'** character are interpreted as comments and are ignored by both the shell and grid-engine as the script is submitted and executed. Lines preceded by the **'#$'** characters are interpreted as grid-engine directives - these options are parsed when the jobscript is submitted, before the script itself is run on any system. Directives can be used to control how grid-engine schedules the job to be run on one or more compute nodes and how you can be notified of the status of your job.

The following common directives are supported by grid-engine:


Copyright © 2011 Alces Software Ltd All Rights Reserved

| *Directive* | *Description* | *qsub example* |
|---|---|---|
| `-a [[CC]]YY]MMDDhhmm[.SS]` | Defines the date and time when a job can be executed. | `-a 201131021830` |
| `-ar ar_id` | Assigns the job to the advanced reservation with ID **`ar_id`** | `-ar 412` |
| `-b y[es]\|n[o]` | Allows a user to indicate that the command provided to qsub is a binary instead of a script. | `-b y mybinary.bin` |
| `-cwd` | Instructs grid-engine to execute the job from the current working directory. | `-cwd` |
| `-display` | Allows grid-engine to configure the remote X display for a graphical application. | `-display imac:3` |
| `-dl [[CC]]YY]MMDDhhmm[.SS]` | Available for users allowed to submit deadline jobs to indicate the final date and time there jobs can be run. | `-dl 201131021830` |
| `-e path` | Indicates the path to be used for standard error output streams. | `-e /users/bob/output` |
| `-hard` | Specifies that jobs submitted with resource requirements must fully satisfy these requirements before they can run. | `-hard` |
| `-hold_jid <jobid>` | Defines that the submitted job must wait before executing until all jobs in the comma-separated **`<jobid>`** list have completed. | `-hold_jid 232,234` |
| `-i <file>` | Specifies that the input stream to the job should be taken from the file **`<file>.`** | `-i /users/bob/input.22` |
| `-j y[es]\|n[o]` | Instructs the scheduler to merge the stdout and stderr streams from the job into a single file. By default, separate files will be created for each stream. | `-j y` |
| `-l resource=value` | Specifies that the job requires a particular resource to be able to run. The **`qconf -sc`** command shows the configured resources available for selection. | `-l exclusive=true` |
| `-m b\|e\|a\|s\|n` | Instructs the scheduler the send email to notify the user when the job:<br>b - begins<br>e - ends<br>a - is aborted<br>s - is suspended<br>n - never email for this job | `-m beas` |
| `-M user[@host]` | Specifies the email address to use to notify users of job status. | `-M myuser` |
| `-notify` | Request that the scheduler sends warning signals (**`SIGUSR1`** and **`SIGUSR2`**) to running jobs prior to sending the actual **`SIGSTOP`** or **`SIGKILL`** messages at | `-notify` |


Copyright © 2011 Alces Software Ltd All Rights Reserved

| *Directive* | *Description* | *qsub example* |
|---|---|---|
| | termination time. | |
| `-now y[es]\|n[o]` | Signifies that interactive jobs run using qsub, qsh, qlogin or qrsh should be scheduled and run immediately or not at all. | `-now y` |
| `-N name` | Allows users to set the name to be used to identify the job. If this parameter is not specified, the job is given the same name as the jobscript. | `-N BobVMDjob4` |
| `-o path` | Indicates the path to be used for standard output streams. | `-o /users/bob/output` |
| `-p priority` | Allows a user to request that their jobs are run with lower than normal priority. Valid priorities are 0 (default) to -1023 (lowest priority). | `-p -1023` |
| `-pe name nodes` | Requests the named parallel environment and number of nodes for the job to be run over. | `-pe mpi-verbose 8` |
| `-q queue` | The scheduler queue to which the job should be submitted. If omitted, the scheduler automatically determines the correct queue to use based on the job type. | `-q serial.q` |
| `-r y[es]\|n[o]` | If set to yes, this parameter causes the scheduler to automatically re-run the job if it fails during execution. | `-r y` |
| `-soft` | Specifies that jobs submitted with resource requirements can still be run even if the requested resources are not available. If this parameter is not specified, the scheduler defaults to **hard** requirements. | `-soft` |
| `-sync y[es]\|n[o]` | Causes the **qsub** command to wait for the job to complete before exiting. | `-sync y` |
| `-t first-last` | Submits a job task array starting at task **first** and ended with task **last**. | `-t 1-1000` |
| `-verify` | Displays results of a dry-run submission without actually submitting the job for execution. | `-verify` |
| `-V` | Exports all current environment variables to the job. | `-V` |
| `-wd path` | Sets the working directory of the job | `-wd /users/bob/app2` |

Grid-engine directives may also be specified at job submission time as parameters to the **qsub** command; for example:

```
[user@login01 ~]$ qsub -j y -V -cwd ./my-serial-job.sh
```

Please note that grid-engine automatically implements a *walltime* setting per job queue which controls the maximum amount of time a user job may be allowed to execute. See your system administrator for more


Copyright © 2011 Alces Software Ltd All Rights Reserved

information on the maximum time limits enforced for the queues configured on your cluster.

## 8.3. GRID-ENGINE PSEUDO ENVIRONMENT VARIABLES

To assist users when writing job scripts, grid-engine automatically creates a number of environment variables that may be referenced in the script. These are:

| Variable name | Description |
|---|---|
| **$HOME** | The user's home directory on the execution machine |
| **$USER** | The user ID of the job owner |
| **$JOB_ID** | The current job ID |
| **$JOB_NAME** | The current job name (may be set via the **-N** directive) |
| **$HOSTNAME** | The name of the execution host |
| **$TASK-ID** | The array job task index number |

Other environment variables may be set manually by the user before submitting the job, or by loading environment modules containing the variables required. Users must remember to use the **-V** directive to export their environment variables to submitted jobs if they have manually configured environment variables.

## 8.4. JOB OUTPUT FILES

By default, grid-engine will collect any information output to the *stdout* or *stderr* channels by your jobscript and write it to files named **jobscript.o** and **jobscript.e** (where "*jobscript*" is the filename of the submitted jobscript). The default location of these files is your home directory, but this can be modified using the **-cwd**, **-o** or **-e** directives. If the **-j y** directive is also specified, *stdout* and *stderr* streams are both written to a single file named **jobscript.o** (or to the location specified by the **-o** directive).

## 8.5. SUBMITTING NON-INTERACTIVE JOBS VIA QSUB

To submit a batch or serial job, use the "**qsub**" command;

```
[user@login01 ~]$ cat my-serial-job.sh
#!/bin/bash
echo "I am a serial job!"
sleep 10

[user@login01 ~]$ qsub my-serial-job.sh
```

Your job will be held in the serial queue until a compute node is available to run it. Some clusters have specific queues configured to run different types of jobs; use the "**qstat -g c**" command to view the queues configured on your cluster:


Copyright © 2011 Alces Software Ltd All Rights Reserved

```
[user@login01 ~]$ qstat -g c
CLUSTER QUEUE                   CQLOAD   USED    RES  AVAIL  TOTAL aoACDS  cdsuE
--------------------------------------------------------------------------------
dev-multiway.q                    0.06      0      0     32     48     16      0
dev-parallel.q                    0.06      1      0      4      6      1      0
dev-serial.q                      0.06      1      0     39     48      8      0
multiway.q                        0.79     32      0    160   1000    760     48
parallel.q                        0.79     86      0     20    125     16      6
serial.q                          0.79     30      0    173   1000    752     48
smp.q                             0.06      4      0     60     64      0      0
```

Use the **-q <queuename>** directive to submit your job to a particular queue as directed by your local system administrator:

```
[user@login01 ~]$ qsub -q serial.q my-serial-job.sh
```

## 8.6. VIEWING THE STATUS OF A SUBMITTED JOB

The grid-engine scheduler allows users to view the status of the jobs they have submitted. The **qstat** command displays the status of all the jobs submitted by the user:

```
job-ID  prior   name       user     state submit/start at     queue                slots
-----------------------------------------------------------------------------------------
   321 0.55500 openmpi.sh  alces-user  r   01/06/2011 10:04:43 parallel.q@node01.lcl  16
   331 0.22000 sleepjob.sh alces-user  r   01/06/2011 11:44:20 serial.q@node00.lcl     1
   332 0.12000 sleepjob.sh alces-user  qw  01/06/2011 11:45:10                         1
   334 0.12000 sleepjob.sh alces-user  qw  01/06/2011 11:48:44                         1
   335 0.12000 sleepjob.sh alces-user  qw  01/06/2011 11:48:52                         1
```

The job state can be marked as one or more of the following:

| Status code | Job state | Description |
|---|---|---|
| **d** | **deleted** | A user or administrator has requested that the job should be deleted from the queuing system |
| **E** | **Error** | The job is in error status. Use the **"-explain E"** option to **qstat** for more information. |
| **h** | **hold** | The job has been set to hold by a user or administrator |
| **r** | **running** | The job is running |
| **R** | **Restarted** | The job has been restarted |
| **s** | **suspended** | The job has been suspended and is not currently running |
| **S** | **Suspended** | The job is currently being suspended |
| **t** | **transferring** | The job is being transferred to an execution host to be run |
| **q** | **queued** | The job is queued for execution |
| **w** | **waiting** | The job is waiting for resources to be available |


Copyright © 2011 Alces Software Ltd All Rights Reserved

By default, the **qstat** command only shows jobs belonging to the user executing the command. Use the **"qstat -u '*'"** command to see the status of jobs submitted by all users.

The **qstat -f** command provides more detail about the scheduler system, also listing the status of each queue instance on every execution host available in your cluster. Queues are listed with the following status:

| Status code | Queue state | Description |
|---|---|---|
| **a** | **alarm (load)** | A queue instance has exceeded its pre-configured maximum load threshold |
| **c** | **configuration error** | A queue instance has a configuration error - contact your system administrator for assistance |
| **d** | **disabled** | A queue instance has been temporarily disabled by a system administrator |
| **o** | **orphaned** | The indicated queue instance has been de-configured, but jobs are still running using queue resources. |
| **s** | **suspended** | The queue instance has been suspended. |
| **u** | **unknown** | The scheduler has lost contact with the machine hosting the queue instance. |
| **A** | **alarm (suspend)** | The queue instance has exceeded its suspension threshold |
| **C** | **Calendar suspended** | The queue has been automatically suspended via the built in calendar facility. Contact your system administrator for information on the configured calendar policies for your site. |
| **D** | **Calendar disabled** | The queue has been automatically disabled via the built in calendar facility. Contact your system administrator for information on the configured calendar policies for your site. |
| **E** | **Error** | The scheduler was unable to contact the shepherd process on the machine hosting this queue instance. Contact your system administrator for assistance. |
| **S** | **Subordinated** | This queue instance has been suspended via subordination to another queue. |

## *8.7. SUBMITTING A PARALLEL JOB*

Your cluster has also been configured with a parallel queue suitable for running MPI jobs across multiple nodes. By default, this queue has been configured with a parallel environment for the MPI environments available on your cluster. To submit a parallel job via the cluster scheduler, users can create a job script and submit it with the *qsub* command, using the **-pe name <slots>** grid-engine directive:


Copyright © 2011 Alces Software Ltd All Rights Reserved

```
[user@login01 /]# cat jobscript.sh
#!/bin/bash
#$ -V -cwd
#$ -pe mpi 4
mpirun -np 8 ./benchmark18.bin

[user@login01 /]# module load mpi/intel/openmpi/1.4

[user@login01 /]# qsub ./jobscript.sh
```

The **mpirun** command included in the jobscript above submits an OpenMPI job with 8 processes - the MPI machinefile is automatically generated by grid-engine and passed to the MPI without needing further parameters. The **-pe mpi 4** directive instructs the scheduler to submit the jobscript to the **MPI** parallel environment using 4 node slots (using 2 processes per node).

There are 2 parallel environments set up on your cluster by default:

- **mpi** - default MPI parallel environment
- **mpi-verbose** - submits the job wrapped in job submission information such as date submitted and queue information.

## 8.8. SUBMITTING INTERACTIVE JOBS:

Grid-engine also allows interactive jobs to be scheduled to run on the compute nodes of your cluster. This method can be used to allow multiple users to run command-line and graphical applications across cluster compute nodes, fairly sharing resources between other interactive applications and batch jobs.

The **"qrsh <binary>"** command can be used to schedule and launch an application, or when invoked without an application name, to launch an interactive shell session on an available compute node. The *qrsh* session is scheduled on the next available compute node on the default interactive queue, or a specific queue specified with the **"-q <queuename>"** parameter.

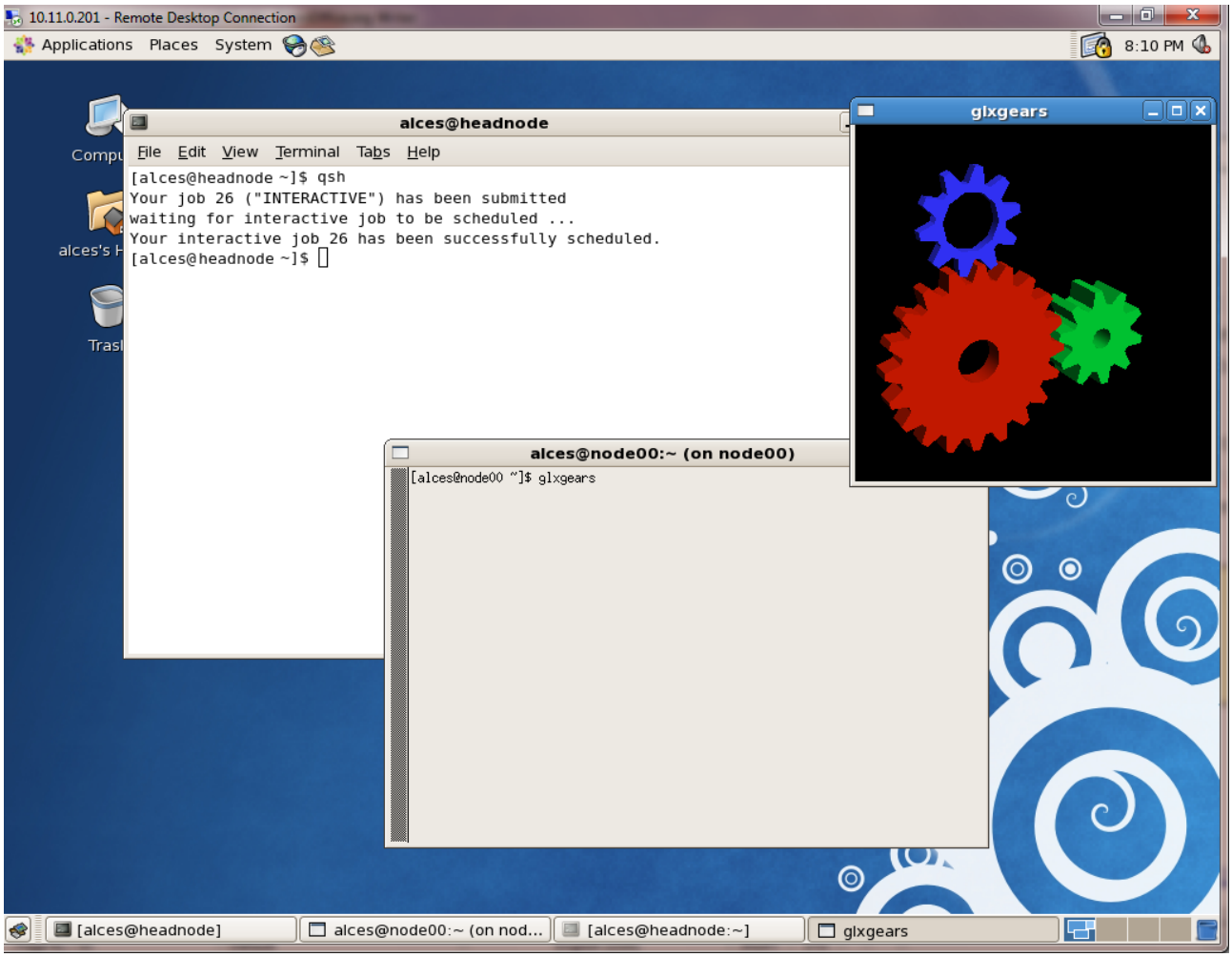
```
[user@login01 /]$ uptime
 03:59:23 up 32 days,  19 users,  load average: 4.32, 5.43, 5.51

[user@login01 /]$ qrsh

[user@node12 ~]$ uptime
 03:59:25 up 58 mins,  1 user,  load average: 0.00, 0.00, 0.00

[user@node12 ~]$
```

The **"qrsh -V xterm"** command can be used to schedule and launch an interactive xterm session on a compute node. The *qrsh* session is scheduled on the next available compute node on the default interactive queue, or a specific queue specified with the **"-q <queuename>"** parameter. The graphical display for your application or *xterm* session will be displayed on the system identified by the **DISPLAY** environment variable setting when *qrsh* was invoked (usually your local graphical terminal). Although graphical applications are typically executed from an interactive graphical desktop session (via *Remote Desktop* or *NX)*, advanced users can direct the graphical display to a workstation outside the cluster by


Copyright © 2011 Alces Software Ltd All Rights Reserved

setting the **DISPLAY** variable before running *qrsh*.

The example below shows how the **qsh** command launches an xterm session on an available compute node. When the graphical **glxgears** application is started, the output window is automatically displayed in the originating *Remote Desktop* graphical desktop window.




## 8.9. SUBMITTING AN ARRAY OF JOBS:

A common problem is that you have a large number of jobs to run, and they are largely identical in terms of the command to run. For example, you may have 1000 data sets, and you want to run a single program on them using the cluster. A quick solution is to generate 1000 separate jobscripts, and submit them all to the queue. This is not efficient, neither for you nor for the scheduler master node.

Grid-engine allows users to submit a single job with a number of separate tasks; these are scheduled with a single job ID (making it simple to track, prioritize or cancel all the jobs) with a number of separate tasks. When the example jobscript below is submitted, an array of 10,000 tasks will be generated and executed on the queue selected.


Copyright © 2011 Alces Software Ltd All Rights Reserved

```
#!/bin/bash
# Export our current environment (-V) and current working directory (-cwd)
#$ -V -cwd
# Tell SGE that this is an array job, with "tasks" to be numbered 1 to 10000
#$ -t 1-10000
# When a single command in the array job is sent to a compute node, its task number is
# stored in the variable SGE_TASK_ID, so we can use the value of that variable to get
#  the results we want:

~/programs/program -i ~/data/input.$SGE_TASK_ID -o ~/results/output.$SGE_TASK_ID
```

The script can be submitted as normal with the **qsub** command and is displayed by grid-engine as a single job with multiple parts:

```
[user@login01 ~]$ qsub -q serial.q array_job.sh
Your job-array 32.1-2:1 ("array_job.sh") has been submitted

[user@login01 ~]$ qstat
job-ID  prior   name       user         state submit/start at     queue
slots ja-task-ID
-----------------------------------------------------------------------------------------
     32 0.55500 array_job. user s        qw   03/27/2010 12:08:02 1 1-10000:1

[user@login01 ~]$
```

## 8.10. JOB DEPENDENCIES:

Users submitting jobs can indicate that grid-engine should wait before starting their job until a previous job or list of jobs has been successfully completed. If the proceeding job is an array job, grid-engine will wait until all array tasks have been completed. The following example may be used to submit a new job called **my_job.sh** that will run when the job with ID number 448 has been completed:

```
[user@login01 ~]$ qsub -q serial.q -hold 448 my_job.sh
Your job 512 ("my_job.sh") has been submitted

[user@login01 ~]$ qstat
job-ID  prior   name       user         state submit/start at     queue
slots ja-task-ID
-----------------------------------------------------------------------------------------
    512 0.55500 my_job. user s        qw   03/29/2010 10:18:44    1

[user@login01 ~]$
```

## 8.11. DELETING A SUBMITTED JOB

Users can delete their own jobs from the scheduler system using the **qdel** command. Jobs which have not yet been scheduled to run will be removed from the queuing system without ever running. Jobs which are already running, or have partially run and been suspended, will be sent **SIGSTOP** and **SIGTERM** signals to stop them executing. Users may specify one or more jobs on the command line, separated by spaces, e.g:

```
[user@login01 ~]$ qdel 324 325
user has deleted job 324
user has deleted job 325
[user@login01 ~]$
```

The job ID was printed to your screen when the job was submitted; alternatively, you can use the **qstat** command to retrieve the number of the job you want to delete.


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 9. TORQUE / OPENPBS CLUSTER SCHEDULER

## 9.1. OVERVIEW OF TORQUE / OPENPBS

Originally known as the Open Portable Batch System (OpenPBS) and later re-released as the Terascale Open-Source Resource and QUEue manager (TORQUE), this scheduler is capable of coordinating and executing large numbers of serial, parallel and interactive jobs across a large HPC cluster. Torque is particularly well suited for integration with workload managers, allowing multiple clusters to be loosely connected together for improved multi-site job scheduling. Even recent releases of the scheduler are often still referred to in the HPC community as both "torque" and "PBS" as well as "OpenPBS".

A typical torque installation requires a *Mother-ship* server (normally the headnode or a cluster service node), one or more submit hosts from where users can submit jobs (typically a login or headnode server) and a number of execution hosts where the jobs are run via the *PBSmom* daemon. The process for running a job through torque is:

- Prepare the application or binary file to run on a cluster node
- Create a **jobscript** to run the application with the required parameters
- Select the torque directives required to control how your job is run
- Submit the jobscript to torque from the cluster login or master node

The steps below indicate how to perform these steps to submit different types of job to the scheduler.

## 9.2. JOB SCRIPTS AND TORQUE DIRECTIVES

A **jobscript** usually takes the form of a simple shell script containing a list of tasks to perform in order to run your job. These may include simple commands (e.g. printing a status message, copying or moving files, etc.) or calling a binary application to execute with parameters. An example jobscript is shown below:

```
#!/bin/bash
# This is an example job script – these lines are comments
# PBS directives are shown below

# Configure the resources needed to run my job
#PBS -l mem=700mb,walltime=1:30:00
# Merge stdout and stderr
#PBS -j oe

echo "Job starting at `date`"
~/test/myapplication -i ~/data/intputfile4
```

Lines preceded by a **'#'** character are interpreted as comments and are ignored by both the shell and torque as the script is submitted and executed. Lines preceded by **'#PBS'** are interpreted as torque directives - these options are parsed when the jobscript is submitted, before the script itself is run on any system. Directives can be used to control how torque schedules the job to be run on one or more compute nodes and how you can be notified of the status of your job.


Copyright © 2011 Alces Software Ltd All Rights Reserved

The following common directives are supported by torque:

| *Directive* | *Description* | *qsub example* |
|---|---|---|
| `-d <dir>` | The working directory to star the job in | `-d /users/myuser/jobs` |
| `-j oe` | Merge the standard error and standard error streams for the submitted job. Parameter **"oe"** merges both streams into the output file; **"eo"** merges into the error file. | `-j oe` |
| `-o <file>` | Instructs the scheduler to redirect the standard output stream to the given filename. | `-o ~/outputs/job3.out` |
| `-e <file>` | Instructs the scheduler to redirect the standard error stream to the given filename. | `-e ~/outputs/job3.err` |
| `-N <name>` | Allows users to set the name to be used to identify the job. If this parameter is not specified, the job is given the same name as the jobscript. | `-N MyJobName4` |
| `-m b\|e\|a\|n` | Instructs the scheduler the send email to notify the user when the job:<br>b - begins<br>e - ends<br>a - is aborted<br>n - never email for this job | `-m abe` |
| `-M user[@host]` | Specifies the email address to use to notify users of job status. | `-M myuser@sdu.ac.uk` |
| `-q queue\|host` | The scheduler queue to which the job should be submitted. If omitted, the scheduler automatically determines the correct queue to use based on the job type. | `-q serial.q`<br>*or*<br>`-q node34` |
| `-V` | Exports all current environment variables to the job. | `-V` |
| `-t first-last`<br>*or*<br>`-t range` | Submits a job task array starting at task **first** and ended with task **last**. Alternatively, if one number is specified, a task for all numbers starting at zero to the specified number are started. | `-t 0-999`<br>*(is equivalent to)*<br>`-t 1000` |
| `-I` | Specifies that the job should be run interactively | `-I` |
| `-l nodes <count>:ppn=<X>:\|gpus=<Y>\|` | Specifies the number of nodes to be reserved for exclusive use by the job. The *ppn* parameter specifies how many processes should be started on each node. The *gpus* parameter allows users to reserve GPGPU devices as well as host CPU cores. | `-l nodes=1:ppn=1:gpus=1`<br>*(1 core in total)*<br><br>`-l nodes=2:ppn=4`<br>*(8 cores in total)*<br><br>`-l nodes=10:ppn=12`<br>*(120 cores in total)* |
| `-l mem=<size>` | Indicates the maximum amount of physical memory used by the job - this value is ignored if the job | `-l mem=500mb` |


Copyright © 2011 Alces Software Ltd All Rights Reserved

| Directive | Description | qsub example |
|---|---|---|
| | runs on more than one node. The *size* parameter may be specified in the following units:<br>`b or w - 1 byte or word`<br>`kb or kw - 1024 bytes (kilobyte)`<br>`mb or mw - 1024 KB (megabytes)`<br>`gb or gw - 1024 MB (gigabytes)`<br>`tb or tw - 1024 GB (terabytes)` | |
| `-l procs` | The number of processors to be allocated to a job. The processors can come from one or more qualified node(s). | `-l procs=2` |
| `-l walltime=hh:mm:ss` | The maximum amount of real time during which the job can be in the running state. | `-l walltime=20:30:00` |
| `-l <other>` | Request a generic resource configured for your site. Contact your system administrator for more information on the resource types available to you. | `-l other=matlab` |

Torque directives may also be specified at job submission time as parameters to the **qsub** command; for example:

```
[user@login01 ~]$ qsub -j oe -l nodes=1 ./my-serial-job.sh
```

Please note that torque automatically implements a maximum *walltime* setting per job queue which controls the maximum amount of time a user job may be allowed to execute. See your system administrator for more information on the maximum time limits enforced for the queues configured on your cluster.

## 9.3. TORQUE PSEUDO ENVIRONMENT VARIABLES

To assist users when writing job scripts, torque automatically creates a number of environment variables that may be referenced in the script. These are:

| Variable name | Description |
|---|---|
| **$PBS_JOBNAME** | The current job name (may be set via the **-N** directive) |
| **$PBS_O_WORKDIR** | The job's submission directory |
| **$PBS_O_HOME** | The home directory of the submitting user |
| **$PBS_O_LOGNAME** | The username of the submitting user |
| **$PBS_O_JOBID** | The torque job ID number |
| **$PBS_O_HOST** | The execution host on which the jobscript is run |
| **$PBS_QUEUE** | The torque queue name the jobscript was submitted to |
| **$PBS_NODEFILE** | A file containing a line delimited list of nodes allocated to the job |
| **$PBS_O_PATH** | The PATH variable used within a jobscript |
| **$PBS_ARRAYID** | The array index number for a task job |


Copyright © 2011 Alces Software Ltd All Rights Reserved

Other environment variables may be set manually by the user before submitting the job, or by loading environment modules containing the variables required in the jobscript.

## 9.4. JOB OUTPUT FILES

By default, torque will collect any information that is output to the *stdout* or *stderr* channels by your jobscript and write it to files named **jobscript.o** and **jobscript.e** (where *jobscript* is the filename of the jobscript submitted). Note that the scheduler only copies output information to this location when the job ends - while the job is still running, output is buffered on the node(s) where the job is running.

The default location of these files is the job working directory, but this can be modified using the **-o** or **-e** directives. If the **-j oe** directive is also specified, *stdout* and *stderr* streams are both written to a single file named **jobscript.o** (or to the location specified by the **-o** directive).

## 9.5. SUBMITTING NON-INTERACTIVE JOBS VIA QSUB

To submit a batch or serial job, use the "**qsub**" command; torque will reply with your job ID:

```
[user@login01 ~]$ cat my-serial-job.sh
#!/bin/bash
#PBS -l walltime=1:00,node=1
echo "I am a serial job!"
sleep 10

[user@login01 ~]$ qsub my-serial-job.sh
193.headnode.cluster.local
[user@login01 ~]$
```

Your job will be held in the serial queue until a compute node is available to run it. Some clusters have specific queues configured to run different types of jobs; use the "**qstat -q**" command to view the queues configured on your cluster:

```
[user@login01 ~]$ qstat -q
server: master.cm.cluster
Queue            Memory CPU Time Walltime Node Run Que Lm State
---------------- ------ -------- -------- ---- --- --- -- -----
serial.q           --      --    48:00:00  --   4   8  --  E R
parallel.q         --      --    96:00:00  --   1   0  --  E R
                                               ----- -----
                                                 5   8
[user@login01 ~]$
```

Queues are listed with the following state:

| Status code | Queue state | Description |
|---|---|---|
| **E** | **Enabled** | The queue is enabled to run jobs |
| **D** | **Disabled** | The queue is disabled and will not schedule new jobs to start. |
| **R** | **Running** | The queue is currently running jobs |
| **S** | **Stopped** | The queue is stopped and is not running jobs. |


Copyright © 2011 Alces Software Ltd All Rights Reserved

Use the **-q <queuename>** directive to submit your job to a particular queue as directed by your local system administrator:

```
[user@login01 ~]$ qsub -q serial.q my-serial-job.sh
```

## 9.6. VIEWING THE STATUS OF A SUBMITTED JOB

The torque scheduler allows users to view the status of the jobs they have submitted. The **"qstat -an"** command displays the status of jobs submitted by a user, including which nodes jobs are running on:

```
headnode.cluster:                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname         SessID NDS   TSK Memory Time  S Time
--------------- -------  -------  --------------- ------ ----- --- ------ ----- - -----
199.headnode.wiz  alces  serial.q testjob2.sh      5549     1   1  700mb 00:24  R   --
   node00/0
200.headnode.wiz  alces  serial.q testjob2.sh      5632     1   1  700mb 00:24  R   --
   node00/1
```

The job state ("**S**") can be marked as one or more of the following:

| Status code | Job state | Description |
|---|---|---|
| **W** | **Waiting** | The job is waiting for resources to be available |
| **T** | **Transferring** | The job is being transferred to a new location |
| **S** | **Suspended** | The job has been suspended and is not currently running |
| **R** | **Running** | The job is running |
| **Q** | **Queued** | The job is queued for execution |
| **H** | **Hold** | The job has been held and is not running |
| **E** | **Exiting** | The job is exiting after having been run |
| **C** | **Completed** | The job has completed execution |

By default, the **qstat** command only shows jobs belonging to the user executing the command. Use the **"qstat -u '*'"** command to see the status of jobs submitted by all users.

## 9.7. SUBMITTING A PARALLEL JOB

Your cluster has also been configured to run MPI jobs across multiple nodes. When submitting a parallel job, the chosen MPI implementation must be provided with a list of nodes to prepare for the job execution. Torque generates this list when a job is scheduled to run and will automatically pass it to OpenMPI when **mpirun** is called. For other MPI implementations that cannot obtain the list of hosts directly from torque, users must use the variable **$PBS_NODEFILE** to send the list to **mpirun** via the **-machinefile** parameter. The example below demonstrates how call **mpirun** in a jobscript when using OpenMPI:


Copyright © 2011 Alces Software Ltd All Rights Reserved

```
[user@login01 /]# cat jobscript-OpenMPI.sh
#!/bin/bash
# Request 4 nodes each with 2 CPUs each
#PBS -l nodes=4:ppn=2
# Request maximum runtime of 12 hours
#PBS -l walltime=12:00:00
# -d /users/myuser/benchmark18
# enable modules environment
. /etc/profile.d/modules.sh
module load mpi/intel/openmpi/1.4
mpirun benchmark18.bin --rasteronly --verbose

[user@login01 /]# qsub ./jobscript-OpenMPI.sh
```

The **mpirun** command included in the jobscript above submits an OpenMPI job with 8 processes - the MPI machine-file is automatically generated by torque and passed to the MPI. The **"-l nodes=4:ppn=2"** torque directive requests that the job is scheduled on 4 nodes, using 2 processes per node.

When using another MPI that does not fully integrate with PBS, your jobscript must provide the **"-machinefile"** and **"-np <processes>"** parameters to **mpirun** to allow the MPI to prepare the necessary nodes for the job; for example:

```
[user@login01 /]# cat jobscript-IntelMPI.sh
#!/bin/bash
# Request 4 nodes each with 2 CPUs each
#PBS -l nodes=4:ppn=2
# Request maximum runtime of 12 hours
#PBS -l walltime=12:00:00
# -d /users/myuser/benchmark18
# enable modules environment
. /etc/profile.d/modules.sh
module load mpi/intel/intelmpi/2.3
mpirun -np 8 -machinefile $PBS_NODEFILE benchmark18.bin --rasteronly --verbose

[user@login01 /]# qsub ./jobscript-IntelMPI.sh
```

## 9.8. EXECUTING AN Interactive JOB

Torque also allows interactive jobs to be scheduled to run on the compute nodes of your cluster. This method can be used to allow multiple users to run command-line and graphical applications across cluster compute nodes, fairly sharing resources between other interactive applications and batch jobs.

The **"-I"** directive can be used to schedule and launch an application interactively, or when invoked without an application name, to launch an interactive shell session on an available compute node. Use the **"-X"** directive to forward your X-display to your current desktop and the **"-x"** if the command you wish to run is a binary rather than a jobscript. For example:

```
[user@login01 /]$ uptime
 03:59:23 up 32 days,  19 users,  load average: 4.32, 5.43, 5.51
[user@login01 /]$ qsub -I -X
4873.login01.cluster.local

[user@node12 ~]$ uptime
 03:59:25 up 58 mins,  1 user,  load average: 0.00, 0.00, 0.00
[user@node12 ~]$
```


Copyright © 2011 Alces Software Ltd All Rights Reserved

The **"qsub -I -X -x xterm"** command can be used to schedule and launch an interactive xterm session on a compute node; the session is scheduled on the next available compute node on the default interactive queue. The graphical display for your application or *xterm* session will be displayed on the system identified by the **DISPLAY** environment variable setting when the job was invoked (usually your local graphical terminal). Although graphical applications are typically executed from an interactive graphical desktop session (via *Remote Desktop* or *NX),* advanced users can direct the graphical display to a workstation outside the cluster by setting the **DISPLAY** variable before submitting the job request.

## 9.9. SUBMITTING AN ARRAY OF JOBS:

A common problem is that you have a large number of jobs to run, and they are largely identical in terms of the command to run. For example, you may have 1000 data sets, and you want to run a single program on them using the cluster. A quick solution is to generate 1000 separate jobscripts, and submit them all to the queue. This is not efficient, neither for you nor for the scheduler master node.

Torque allows users to submit a single job with a number of separate tasks; these are scheduled with a single job ID (making it simple to track, prioritize or cancel all the jobs) with a number of separate tasks. When the example jobscript below is submitted, an array of 10,000 tasks will be generated and executed on the queue selected.

```
#!/bin/bash
# Request maximum runtime (per task job) of 1 hour
#PBS -l walltime=1:00:00

# Merge stdout and stderr, and set location of output files
#PBS -j oe -o /users/user/jobs/output/

# Tell PBS that this is an array job, with "tasks" to be numbered 1 to 10000
#PBS -t 1-10000

# When a single command in the array job is sent to a compute node, its task number is
# stored in the variable PBS_ARRAYID, so we can use the value of that variable to get
#  the results we want:

~/programs/program -i ~/data/input.$PBS_ARRAYID  -o ~/results/output.$PBS_ARRAYID
```

The script can be submitted as normal with the **qsub** command and is displayed by torque as a single job with multiple parts. Each separate task will be scheduled and executed separately, each with its own output and error stream (depending on the directives used). Use the **"qstat -ant"** command to view the status of individual tasks:

```
[user@headnode jobs]$ qstat -ant
headnode.cluster.local
                                                                  Req'd  Req'd   Elap
Job ID           Username Queue    Jobname          SessID NDS   TSK Memory Time  S Time
---------------- -------- -------- ---------------- ------ ----- --- ------ ----- - ----
232[1].headnode   user    serial.q taskjob.sh-25    9598   --    --  700mb 00:24 R 00:00
   node00/1
232[2].headnode   user    serial.q taskjob.sh-26    9608   --    --  700mb 00:24 R 00:00
   node00/0
232[3].headnode   user    serial.q taskjob.sh-39   10438   --    --  700mb 00:24 R   --
   node00/10
232[4].headnode   user    serial.q taskjob.sh-40    1342   --    --  700mb 00:24 Q   --
```


Copyright © 2011 Alces Software Ltd All Rights Reserved

## 9.10. REQUESTING EXCLUSIVE USE OF A COMPUTE NODE

Your local system administrators may allow certain power users to request exclusive use of one or more compute nodes for their jobs. When selected, this *quality-of-service* setting instructs the scheduler not to allow any other jobs to share the same physical compute nods as the exclusive job. Typical uses include when users know in advance that their jobs require exclusive use of a local compute node device such as a hard disk, network interface or GPGPU device. System administrators often restrict the number of jobs which can request exclusive status as it can lead to inefficient use of resources and longer queue times before jobs can be serviced.

To request exclusive use of compute resources, use the **"-x QOS:exclusive"** torque directive when submitting your job:

```
[user@login01 ~]$ qsub -l nodes=8:ppn=8 -x QOS:exclusive ./my-parallel-job.sh
```

## 9.11. DELETING A SUBMITTED JOB

Users can delete their own jobs from the scheduler system using the **qdel** command. Jobs which have not yet been scheduled to run will be removed from the queuing system without ever running. Jobs which are already running, or have partially run and been suspended, will be sent **SIGSTOP** and **SIGTERM** signals to stop them executing. Users may specify one or more jobs on the command line, separated by spaces, e.g:

```
[user@headnode jobs]$ qstat
Job id                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
231.headnode               testjob.sh       user                   0 R serial.q

[user@headnode jobs]$ qdel 231
[user@headnode jobs]$ qstat
[user@headnode jobs]$
```

The job ID was printed to your screen when the job was submitted; alternatively, you can use the **qstat** command to retrieve the number of the job you want to delete, as shown above.


Copyright © 2011 Alces Software Ltd All Rights Reserved

# 10. LUSTRE PARALLEL FILESYSTEM

## 10.1. LUSTRE BACKGROUND

The Lustre parallel filesystem provides scalable, high-performance storage services to multiple client nodes via a range of interconnect technologies. A file stored in a Lustre filesystem is available to all clients mounting the filesystem, similar to filesystems shared with the NFS or CIFS file sharing protocols. No special file handling is required to use the shared filesystem - all POSIX compliant applications and utilities are compatible.

Files stored in a Lustre filesystem are separated into their metadata (file names, sizes, locations, permissions, etc.) and a number of data blocks. File metadata is stored by the Lustre metadata servers (MDS) and block data is stored by object storage servers (OSS). A single shared filesystem requires one MDS (typically deployed as a pair of high-availability servers) and two or more OSS. The usable size and performance of the resulting filesystem depends on the configuration of OSS machines supporting it.

The Lustre filesystem may be accessed via a range of network technologies including gigabit Ethernet, 10-gigabit Ethernet and Infiniband - the only requirement is that all servers accessing the filesystem have network access to all the MDS and OSS machines. Lustre also supports a gateway service which can be used to extend the filesystem to remote clients across a LAN or WAN link. The gateway can also be used to allow filesystem access for remote Ethernet connected clients, irrespective of the local Lustre interconnect type.

## 10.2. QUERYING FILESYSTEM SPACE

The **"lfs df"** command is used to determine available disk space on a Lustre filesystem. It displays the amount of available disk space on the mounted Lustre filesystem and shows space consumption per OST. If multiple Lustre filesystems are mounted, a path may be specified, but is not required. Supported options includes:

| | |
|---|---|
| **-h** | Human-readable print sizes in an easily readable format |
| **-i, --inodes** | Lists inodes instead of block usage |

The **"df -i"** and **"lfs df -i"** commands show the minimum number of inodes that can be created in the filesystem. Depending on the configuration, it may be possible to create more inodes than initially reported by **"df -i"**. Later, **"df -i"** operations will show the current, estimated free inode count. If the underlying filesystem has fewer free blocks than inodes, then the total inode count for the filesystem reports only as many inodes as there are free blocks. This is done because Lustre may need to store an external attribute for each new inode, and it is better to report a free inode count that is the guaranteed, minimum number of inodes that can be created.


Copyright © 2011 Alces Software Ltd All Rights Reserved

```
# lfs df
UUID                 1K-blocks Used      Available Use%    Mounted on
mds-lustre-0_UUID  9174328   1020024   8154304   11%     /mnt/lustre[MDT:0]
ost-lustre-0_UUID  94181368  56330708  37850660  59%     /mnt/lustre[OST:0]
ost-lustre-1_UUID  94181368  56385748  37795620  59%     /mnt/lustre[OST:1]
ost-lustre-2_UUID  94181368  54352012  39829356  57%     /mnt/lustre[OST:2]
filesystem summary:282544104 167068468 39829356  57%     /mnt/lustre

# lfs df -i
UUID                 Inodes   IUsed  IFree   IUse% Mounted on
mds-lustre-0_UUID  2211572  41924  2169648 1%    /mnt/lustre[MDT:0]
ost-lustre-0_UUID  737280   12183  725097  1%    /mnt/lustre[OST:0]
ost-lustre-1_UUID  737280   12232  725048  1%    /mnt/lustre[OST:1]
ost-lustre-2_UUID  737280   12214  725066  1%    /mnt/lustre[OST:2]
filesystem summary:2211572  41924  2169648 1%    /mnt/lustre[OST:2]
```


Copyright © 2011 Alces Software Ltd All Rights Reserved

# APPENDIX A: EXAMPLE PARALLEL JOB EXECUTION

The following example demonstrates how to write a basic MPI program and submit it to run on the cluster via the grid-engine scheduler. Log on to your cluster as a user (not the root account), and make a new directory for your MPI test program:

```
[user@login01 ~]# mkdir ~/mpi_test
[user@login01 ~]# cd ~/mpi_test
```

Create a new file called **"~/mpi_test/test.c"** and enter the following C code to form your MPI program:

```
#include <stdio.h>
#include <mpi.h>
#include <time.h>
#include <string.h>
    int main(int argc, char **argv) {
        char    name[MPI_MAX_PROCESSOR_NAME];
        int     nprocs, procno, len;

        MPI_Init( &argc, &argv );
        MPI_Comm_size( MPI_COMM_WORLD, &nprocs );
        MPI_Comm_rank( MPI_COMM_WORLD, &procno );
        MPI_Get_processor_name( name, &len );
        name[len] = '\0';

        time_t lt;
        lt = time(NULL);
        printf( "Hello !! from %s@%d/%d on %s\n", name, procno, nprocs, ctime(&lt));
        MPI_Barrier( MPI_COMM_WORLD );
        MPI_Finalize();
        return( 0 );
    }
```

The source code for this short program is stored in the **/opt/alces/examples** directory on your HPC cluster for convenience. Use the **modules** command to load the *OpenMPI* environment and compile your C code to make an executable binary called **mpi_test**:

```
[user@login01 ~]# module load mpi/openmpi-1.2.6-gcc
[user@login01 ~]# mpicc -o mpi_test  -O3 test.c
```

Next, create a new file called **"~/mpi_test/sub.sh"** and edit it to contain the following lines to form a grid-engine jobscript:

```
# SGE SUBMISSION SCRIPT
# Specify the parallel environment and number of nodes to use
# Submit to the mpi-verbose parallel environment by default so we can see
# output messages
#$ -pe mpi-verbose 2

#Specify specific SGE options
#$ -cwd -V -j y

#Specify the SGE job name
```


Copyright © 2011 Alces Software Ltd All Rights Reserved

```
#$ -N mpi_test

#Specify the output file name
#$ -o OUTPUT

# A Machine file for the job is available here if you need to modify
# from the SGE default and pass to mpirun. The machine hostnames in use are
# listed in the file separated by new lines (\n) e.g:
# node00
# node01
MACHINEFILE="/tmp/sge.machines.$JOB_ID"

#The mpirun command - specify the number of processes per node and the binary
mpirun -machinefile /tmp/sge.machines.$JOB_ID -np 2 ~/mpi_test/mpi_test
```

Submit a new job to the cluster scheduler that will execute your binary using the grid-engine jobscript. Confirm the job is running with the **qstat** command, and read the output file to view the results of the job execution.

```
[user@login01 mpi_test]$ qsub sub.sh
Your job 11 ("mpi_test") has been submitted
[user@login01 mpi_test]$ qstat -f
queuename                      qtype resv/used/tot. load_avg arch        states
---------------------------------------------------------------------------------
parallel.q@node00.alces-softwa IP    0/1/1          0.06     lx24-amd64
     11 0.55500 mpi_test   user         r     02/02/2009 12:18:49     1
---------------------------------------------------------------------------------
parallel.q@node01.alces-softwa IP    0/1/1          0.04     lx24-amd64
     11 0.55500 mpi_test   user         r     02/02/2009 12:18:49     1
---------------------------------------------------------------------------------
serial.q@node00.alces-software BI    0/0/8          0.06     lx24-amd64    S
---------------------------------------------------------------------------------
serial.q@node01.alces-software BI    0/0/8          0.04     lx24-amd64    S

[user@login01 mpi_test]$ cat OUTPUT
============================================================
SGE job submitted on Wed May 27 04:14:15 BST 2009
2 hosts used
JOB ID: 43
JOB NAME: mpi_test
PE: mpi
QUEUE: parallel.q
MASTER node00.alces-software.org
Nodes used:
node00 node01
============================================================
Job Output Follows:
================================================

Hello !! from node00@2/8 on Wed May 27 04:14:30 2009
Hello !! from node00@0/8 on Wed May 27 04:14:30 2009
Hello !! from node00@4/8 on Wed May 27 04:14:30 2009
Hello !! from node00@6/8 on Wed May 27 04:14:30 2009
Hello !! from node01@1/8 on Wed May 27 04:14:37 2009
Hello !! from node01@3/8 on Wed May 27 04:14:37 2009
Hello !! from node01@5/8 on Wed May 27 04:14:37 2009
Hello !! from node01@7/8 on Wed May 27 04:14:37 2009

================================================
SGE job completed on Wed May 27 04:14:31 BST 2009
================================================
```


Copyright © 2011 Alces Software Ltd All Rights Reserved

# APPENDIX B: EXAMPLE OPENMP JOB COMPILATION

The OpenMP API supports multi-platform shared-memory parallel programming in C/C++ and Fortran. OpenMP is a portable, scalable model with a simple and flexible interface for developing parallel applications on platforms from the desktop to the supercomputer. OpenMP jobs may be executed on standard compute nodes, or used on dedicated SMP nodes, often equipped with larger numbers of processing cores and more system memory.

The following example demonstrates how to write a basic OpenMP program and submit it to run on the cluster via the SGE scheduler. Log in to your cluster as a user (**not** the root account), and make a new directory for your OpenMP test program:

```
[user@login01 ~]# mkdir ~/openmp_test
[user@login01 ~]# cd ~/openmp_test
```

Create a new file called **"~/openmp_test/hello.c"** and enter the following C code to form your OpenMP program:

```
#include <omp.h>
#include <stdio.h>
int main(int argc, char*  argv[]) {
    int id;
    #pragma omp parallel private(id)
    {
         id = omp_get_thread_num();
         printf("%d: Hello World!\n", id);
    }
    return 0;
}
```

Compile your C code to make an executable binary called **hello** and use the **ldd** command to confirm the shared libraries it will use on execution:

```
[user@login01 smp]$ gcc -fopenmp -o hello hello.c
[user@login01 smp]$ ldd hello
    libgomp.so.1 => /usr/lib64/libgomp.so.1 (0x00002b3c9fff9000)
    libpthread.so.0 => /lib64/libpthread.so.0 (0x0000003283e00000)
    libc.so.6 => /lib64/libc.so.6 (0x0000003283200000)
    librt.so.1 => /lib64/librt.so.1 (0x0000003284200000)
    /lib64/ld-linux-x86-64.so.2 (0x0000003282e00000)
```

Next, create a new file called **"~/openmp_test/run-job.sh"** and edit it to contain the following lines to form a grid-engine jobscript:

```
# SGE SUBMISSION SCRIPT
# Submit to the smp-verbose parallel environment by default so we can see
# output messages. Use 8 slots in the parallel environment, corresponding
# to the number of threads to use (one slot = one thread = one CPU core)
#
#$ -V -cwd -pe smp-verbose 8 -j y -o out.$JOB_ID
./hello
```


Copyright © 2011 Alces Software Ltd All Rights Reserved

Submit a new job to the cluster scheduler that will execute your binary using the grid-engine jobscript. Confirm the job is running with the **qstat** command, and read the output file to view the results of the job execution.

```
[user@login01 mpi_test]$ qsub run-job.sh
Your job 11 ("hello") has been submitted
[user@login011 smp]$ qstat -f -q smp.q
queuename                       qtype resv/used/tot. load_avg arch
---------------------------------------------------------------------------
smp.q@smp1.aalces-software.com  IP    0/8/32         0.00     lx24-amd64
     31 0.55500 run-job.sh alces       r     12/01/2009 20:10:27     8
---------------------------------------------------------------------------
smp.q@smp2.alces-software.com  IP    0/0/32         0.00     lx24-amd64

[user@login01 smp]$ cat out.31
============================================================
SGE job submitted on Tue Sep 8 10:10:19 GMT 2009
JOB ID: 31
JOB NAME: run-job.sh
PE: smp-verbose
QUEUE: smp.q
MASTER smp1.alces-software.com
============================================================

** A machine file has been written to /tmp/sge.machines.31 on
smp1.alces-software.com **

============================================================
If an output file was specified on job submission
Job Output Follows:
============================================================

0: Hello World!
1: Hello World!
7: Hello World!
4: Hello World!
2: Hello World!
3: Hello World!
5: Hello World!
6: Hello World!
================================================
SGE job completed on Tue Sep 8 10:10:28 GMT 2009
================================================
```


Copyright © 2011 Alces Software Ltd All Rights Reserved